









26

2018

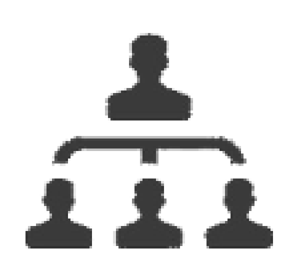
10 Años Aportando al Conocimiento

2018 - Bolvia

http://revista.atixlibre.org
Twitter: @atixlibre
Facebook: facebook.com/Atix.Libre







DIRECCION GENERAL
Esteban Saavedra Lopez



DIAGRAMACION
Jenny Saavedra Lopez
Esteban Saavedra Lopez



REVISION
Jenny Saavedra Lopez



CONTACTO
info@atixlibre.org
http://revista.atixlibre.org





Hoy se celebra el día mundial del Software Libre, o más conocido como el Software Freedom Day, evento que año tras año va promoviendo el movimiento del software libre en el mundo.

Muchos coincidimos que no solo debiese celebrarse una vez al año, sino día a día por que refleja la lucha de un movimiento que nos ha otorgado libertades sobre el software.

Lastimosamente en muchos países, principalmente el nuestro existen muchas instituciones que pregonan hacer uso de software libre, llegando a ser falsos profetas, ya que su posición es netamente política y oportunista; cuando en realidad debiese ser una convicción de libertad tecnológica que es el verdadero sentido de este movimiento.

Es una verdadera lastima que personas que nunca antes promovieron o colaboraron o lucharon por el movimiento del software libre, sean hoy quienes figuren como las cabezas de muchas instituciones de gobierno como si fueran los artífices del movimiento de software libre, sin ni siquiera haber realizado un aporte a este movimiento y más aún no tengan la convicción de hacerlo, sino solamente lo hagan por un postulado político y mezquino.

Pero hoy no es un día de lamentación, sino un día de celebración para todos aquellos que estuvimos apoyando y promoviendo las libertades del software libre desde sus inicios y de seguro estaremos aquí mucho mas tiempo que los oportunistas que solo están de paso.

Salud Freedom Day 2018

Sean bienvenidos a nuestra edición número 26.



Esteban Saavedra L.

Presidente Fundación AtixLibre

Contenido

Número 26 - Septiembre 2018

Software Freedom Day





LXC
Manejo de Linux Containers

Los contenedores Linux son otro paso en la evolución sobre cómo desarrollamos, implementamos y administramos las aplicaciones. Las imágenes de contenedores de Linux proporcionan portabilidad y control de versión, lo que permite garantizar que lo que funciona en la portátil del desarrollador también funciona en los servidores de producción.



Contenedores Linux

Un contenedor de Linux® es un conjunto de procesos separados del resto del sistema, los cuales pueden ejecutarse a partir de una imagen que proporciona todos los archivos necesarios para que funcione correctamente; este aspecto hace que un aplicación sea portátil y consistente indistintamente de la etapa en que se encuentre (desarrollo, prueba, producción).

LXC

- Es una tecnología de virtualización en el nivel de sistema operativo (SO) para Linux. Permite que un servidor físico ejecute múltiples instancias de sistemas operativos aislados, conocidos como Servidores Privados Virtuales (VPS) o Entornos Virtuales (EV).
- No provee de una máquina virtual, más bien provee un entorno virtual que tiene su propio espacio de procesos y redes.

Objetivo

El objetivo de los contenedores Linux es poder desarrollar, desplegar y satisfacer más rápido las demandas de las empresas a medida que surgen. No debe limitarse a pensar que los contenedores solo retienen una aplicación única; los contenedores se pueden aplicar para retener partes de una aplicación o servicios.

Los contenedores son una excelente manera de entregar software y aplicaciones a sus clientes más rápido.

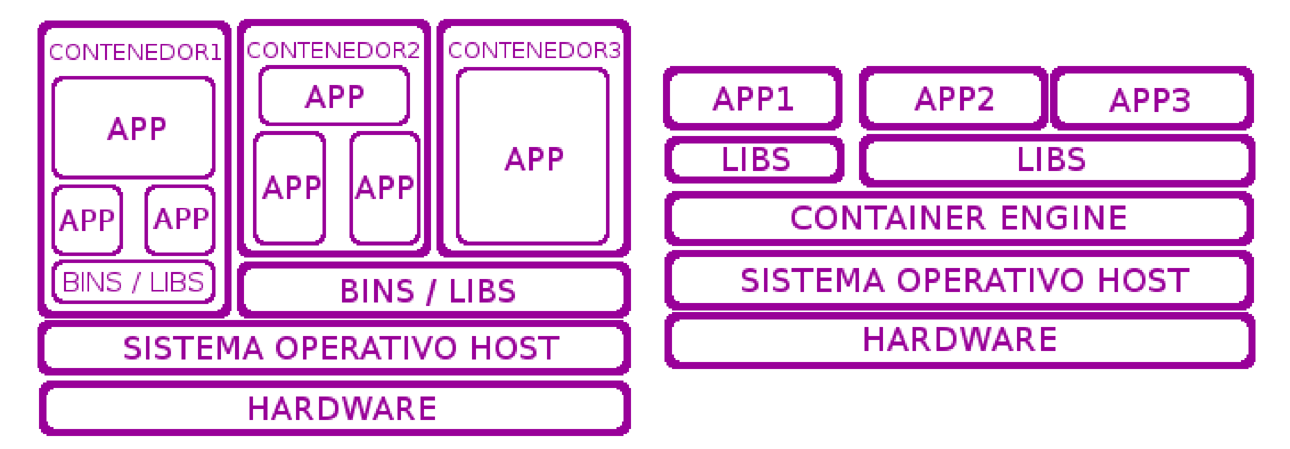


Posibilidades

- Provee un completo entorno de sistema operativo aislado y empaquetado
- Permite empaquetar y aislar aplicaciones con un completo entorno de ejecución
- Provee un entorno ligero y portable
- Ayuda a maximizar el uso de recursos dentro un data center
- Ayuda en el despliegue de diferentes entornos (desarrollo, test, producción)

Clasificación de contenedores

- **A nivel de sistema operativo**: Un sistema operativo completo se ejecuta en un espacio aislado dentro del HOST, compartiendo el mismo kernel, sobre el cual se pueden instalar y configurar distintas servicios.
- **A nivel de aplicación**: Una aplicación o servicio conjuntamente con los procesos mínimos requeridos por la aplicación, se ejecutan dentro de un espacio aislado dentro el HOST.



Nivel Sistema Operativo

Nivel Aplicación

Ventajas

- Los contenedores son ligeros comparados a las máquinas virtuales.
- A diferencia de las máquinas virtuales no requieren capas de emulación (Hardware, Software)
- Los contenedores comparten recursos, mediante espacios de usuario y procesos de forma aislada
- Por su naturaleza de ser ligeros, proporciona la capacidad de ejecutar mayor cantidad de contenedores dentro un HOST
- Su inicio es casi instantáneo en comparación con las máquinas virtuales.
- Su empaquetamiento de entornos completos con sus paquetes requeridos, genera una gran ventaja en su portabilidad.

Instalación

```
$ apt install lxc
```

Verificar la configuración

```
$ lxc-checkconfig
Kernel configuration not found at /proc/config.gz; searching...
Kernel configuration found at /boot/config-4.4.0-134-generic
--- Namespaces ---
Namespaces: enabled
Utsname namespace: enabled
Ipc namespace: enabled
Pid namespace: enabled
User namespace: enabled
Warning: newuidmap is not setuid-root
Warning: newgidmap is not setuid-root
Network namespace: enabled
Multiple /dev/pts instances: enabled

--- Control groups ---
Cgroup: enabled
Cgroup clone_children flag: enabled
Cgroup device: enabled
Cgroup sched: enabled
Cgroup cpu account: enabled
Cgroup memory controller: enabled
Cgroup cpuset: enabled

--- Misc ---
Veth pair device: enabled
Macvlan: enabled
Vlan: enabled
Bridges: enabled
Advanced netfilter: enabled
CONFIG_NF_NAT_IPV4: enabled
CONFIG_NF_NAT_IPV6: enabled
CONFIG_IP_NF_TARGET_MASQUERADE: enabled
CONFIG_IP6_NF_TARGET_MASQUERADE: enabled
CONFIG_NETFILTER_XT_TARGET_CHECKSUM: enabled
FUSE (for use with lxcfs): enabled

--- Checkpoint/Restore ---
checkpoint restore: enabled
CONFIG_FHANDLE: enabled
CONFIG_EVENTFD: enabled
CONFIG_EPOLL: enabled
CONFIG_UNIX_DIAG: enabled
CONFIG_INET_DIAG: enabled
CONFIG_PACKET_DIAG: enabled
CONFIG_NETLINK_DIAG: enabled
File capabilities: enabled

Note : Before booting a new kernel, you can check its configuration
usage : CONFIG=/path/to/config /usr/bin/lxc-checkconfig
```



Archivos de configuración

Los archivos de configuración permiten establecer parámetros, bajo los cuales funcionarán los contenedores, estos archivos se encuentran en:

`/var/lib/lxc/nombre_del_contendor/config`.

Configuración por defecto

```
/etc/lxc/default.conf
lxc.network.type = veth
lxc.network.link = lxcbr0
lxc.network.flags = up
lxc.network.hwaddr = 00:18:5e:xx:xx:xx
```

Reiniciar el servicio

```
$ service lxc-net restart
```

Configuración de red

```
/etc/default/lxc-net
USE_LXC_BRIDGE="true"
LXC_BRIDGE="lxcbr0"
LXC_ADDR="10.0.3.1"
LXC_NETMASK="255.255.255.0"
LXC_NETWORK="10.0.3.0/24"
LXC_DHCP_RANGE="10.0.3.2,10.0.3.254"
LXC_DHCP_MAX="253"
```

Directorio de templates

```
/usr/share/lxc/templates
drwxr-xr-x 2 root root  4096 Aug 27 00:58 .
drwxr-xr-x 6 root root  4096 Aug 27 00:58 ..
-rwxr-xr-x 1 root root 13042 Jun 14  2017 lxc-alpine
-rwxr-xr-x 1 root root 13737 Jun 14  2017 lxc-altlinux
-rwxr-xr-x 1 root root 11156 Jun 14  2017 lxc-archlinux
-rwxr-xr-x 1 root root 11649 Jun 14  2017 lxc-busybox
-rwxr-xr-x 1 root root 29725 Jun 14  2017 lxc-centos
-rwxr-xr-x 1 root root 10374 Jun 14  2017 lxc-cirros
-rwxr-xr-x 1 root root 20171 Jun 14  2017 lxc-debian
-rwxr-xr-x 1 root root 18197 Jun 14  2017 lxc-download
-rwxr-xr-x 1 root root 49693 Jun 14  2017 lxc-fedora
-rwxr-xr-x 1 root root 28259 Jun 14  2017 lxc-gentoo
-rwxr-xr-x 1 root root 13849 Jun 14  2017 lxc-openmandriva
-rwxr-xr-x 1 root root 15910 Jun 14  2017 lxc-opensuse
-rwxr-xr-x 1 root root 41596 Jun 14  2017 lxc-oracle
-rwxr-xr-x 1 root root 11463 Jun 14  2017 lxc-plamo
-rwxr-xr-x 1 root root 19096 Jun 14  2017 lxc-slackware
-rwxr-xr-x 1 root root 26667 Jun 14  2017 lxc-sparclinux
-rwxr-xr-x 1 root root  6822 Jun 14  2017 lxc-sshd
-rwxr-xr-x 1 root root 25703 Jun 14  2017 lxc-ubuntu
-rwxr-xr-x 1 root root 11734 Jun 14  2017 lxc-ubuntu-cloud
```

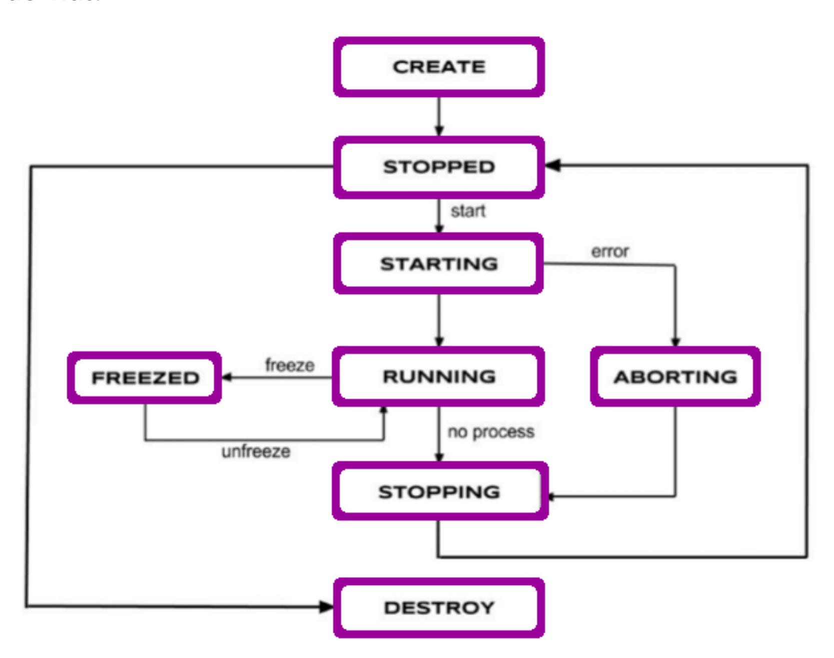


Imágenes y repositorios

LXC posee servidores públicos que contienen imágenes de una gran variedad de distribuciones Linux y que son actualizadas frecuentemente. Estas imágenes sirven como plantillas para crear contenedores y están disponibles para distintas arquitecturas.

Ciclo de vida de un contenedor

Desde el momento que se crea un contenedor, éste puede atravesar por distintas etapas, lo que delimita su ciclo de vida.



Operaciones con contenedores

Son muchas las operaciones que se pueden realizar con los contenedores, las más frecuentes las detallamos a continuación:

lxc-attach	lxc-create	lxc-snapshot
lxc-autostart	lxc-destroy	lxc-start
lxc-cgroup	lxc-device	lxc-start-ephemeral
lxc-checkconfig	lxc-execute	lxc-stop
lxc-checkpoint	lxc-freeze	lxc-top
lxc-clone	lxcfs	lxc-unfreeze
lxc-config	lxc-info	lxc-unshare
lxc-console	lxc-ls	lxc-usernsexec
lxc-copy	lxc-monitor	lxc-wait



Crear un contenedor

Crear un contenedor desde los templates públicos

```
$ lxc-create -t download -n web-server
```

```
Distribution: debian
Release: stretch
Architecture: amd64

Downloading the image index
Downloading the rootfs

Downloading the metadata
The image cache is now ready
Unpacking the rootfs
---
You just created a Debian container (release=stretch, arch=amd64, variant=default)
To enable sshd, run: apt-get install openssh-server
For security reason, container images ship without user accounts
and without a root password.
Use $ lxc-attach or chroot directly into the rootfs to set a root password
or create user accounts.
```

Crear un contenedor desde un templates específico

```
$ lxc-create -t fedora -n web-server -- --release=25
```

Crear un contenedor que contenga ciertos paquetes

```
$ lxc-create -t debian -n stretch-test -- --release stretch --packages=wget,iputils-
ping
```

Operaciones con contenedores

Iniciar un contenedor

```
$ lxc-start -n web-server -d
```

Listado de contenedores

```
$ lxc-ls --fancy
```

Ingresar al contenedor

```
$ lxc-console -n web-server
```

```
$ lxc-attach -n web-server
```

Información del contenedor

```
$ lxc-info -n web-server
```

Detener un contenedor

```
$ lxc-stop -n web-server
```

Crear copias de un contenedor

```
$ lxc-copy --snapshot --name web-server --newname web-server_10
```

Clonar contenedores

```
$ lxc-clone -o mail-server -n servidor-correo
```

Crear un contenedor con auto arranque

```
$ lxc-create --name server-ubuntu --template ubuntu
```

Listar los contenedores con autoarranque

```
$ lxc-autostart --list
```

Configurar todos los contenedores para auto arranque

```
$ lxc-autostart --all
```

Configuración de un contenedor

Para configurar alguna característica específica de un contenedor, es necesario editar el archivo de configuración del contenedor: **`/var/lib/lxc/nombre_del_contendor/config.`**

Inicio desde el arranque

```
lxc.start.auto = 1
```

Demora antes de iniciar

```
lxc.start.delay = 5
```

Prioridad de arranque

```
lxc.start.order = 100
```

Compartir carpetas entre HOST y contenedor

Para poder compartir una carpeta entre el HOST y el contenedor, debemos seguir los siguientes pasos:

HOST

Crear el directorio a compartir

```
mkdir /opt/datos && chmod 7777 /opt/datos
```

Contenedor

Crear el directorio asociado al directorio del HOST

```
mkdir /datos
```

Editar la configuración del contenedor **`/var/lib/lxc/nombre_del_contendor/config`**

```
lxc.mount.entry = /opt/datos datos none ro,bind 0.0
```



Conclusiones

- El uso de contenedores Linux, nos permite la creación de verdaderos entornos de desarrollo o test, dentro de los cuales pueden desplegarse una o varias aplicaciones, con su propia instalación de paquetes y configuración, pero con la posibilidad de poder interactuar entre todas y brindar un conjunto de servicios integrados.
- La creación de entornos simples o múltiples permite probar nuestro proyectos de desarrollo, como si se tratase de un verdadero entorno de producción.
- La posibilidad de poder crear nuestras propias imágenes, nos da la posibilidad de poder distribuir o transportar nuestros entornos de desarrollo o test totalmente personalizados, listos para su uso.

Agradecimiento

Nuevamente mi agradecimiento al Ing. Esteban Saavedra, por haberme impulsado a investigar y conocer el mundo de los contenedores Linux.

Referencias

[1] http://www.linuxcontainers.org





Las aplicaciones se están volviendo más complejas y la demanda por un desarrollo más rápido es cada vez mayor. Esto pone presión en su infraestructura, equipos de TI y procesos. Los contenedores de Linux® ayudan a reducir estos problemas y a iterar más rápido en varios entornos.

Introducción

Los contenedores de Linux son tecnologías que le permiten empaquetar y aislar aplicaciones con todo su entorno de tiempo de ejecución (todos los archivos necesarios para ejecutarse); esto le permite mover su aplicación contenida entre entornos (desarrollo, prueba, producción, etc.) al tiempo que conserva toda la funcionalidad.

Beneficios

- Los contenedores Linux ayudan a reducir conflictos entre sus equipos de desarrollo y operaciones separando áreas de responsabilidad. Los desarrolladores pueden centrarse en sus aplicaciones y el sector de operaciones se puede centrar en la infraestructura. Y debido a que los contenedores Linux se basan en tecnología de open source, usted obtiene el último y mayor avance tan pronto esté disponible.
- Cuando su negocio necesita la máxima portabilidad en varios entornos, el uso de contenedores siempre puede ser la mejor opción.
- Los contenedores permiten garantizar la consistencia en entornos y varios objetivos de implementación, como servidores físicos, máquinas virtuales y nubes privadas o públicas.

¿Qué se puede hacer con los contenedores?

Puede implementar contenedores para una serie de cargas de trabajo y casos prácticos, de pequeños a grandes. Los contenedores proporcionan a su equipo la tecnología subyacente necesaria para un enfoque actual del desarrollo de software, como DevOps y CI/CD (integración continua e implementación continua).

LXD

- Es un hypervisor para contenedores Linux desarrollado por Canonical Ltd. Es un complemento para los contenedores de Linux (LXC) que facilita su uso y añade nuevas posibilidades.
- Brinda una experiencia similar a las máquinas virtuales, pero evitando la sobrecarga en el uso de recursos.

- Representa la próxima generación de gestión de contenedores, basado en imágenes pre-construidas de un gran número de distribuciones Linux.
- Esta construida sobre una simple pero poderosa REST API

Características



Seguro: Permite realizar restricciones en el uso de recursos y acceso a los mismos.



Ligero: Al compartir el mismo kernel del HOST, permite que la ejecución de contenedores sea más liviana.



Escalable: Permite escalar soluciones desde el uso de simples contenedores hasta el manejo de cluster.



Basado en imágenes: Permite crear contenedores basado en imágenes de una variedad de distribuciones Linux.



Uso del mismo kernel: No precisa emulación de ningún tipo ya que hace uso del mismo kernel del HOST



Monitoreo: permite realizar el monitoreo de recursos utilizados.



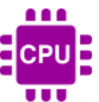
Intuitivo: Por medio de su API, permite crear una experiencia simple, sencilla y clara para su interacción



Gestión de Profiles: Permite la creación y administración de perfiles personalizados de grupos y contenedores individuales.



Mayor Cantidad: Al disponer de una mejor administración y gestión de recursos, permite que se puedan ejecutar una mayor cantidad de contenedores.



Gestión de restricciones: Permite establecer restricciones en el uso de ciertos recursos del contenedor.



Gestión de Dispositivos: Permite adicionar y configurar diversos dispositivos dentro de los contenedores (USB,GPU,NICs,etc)



Gestión de Red: Permite la creación y configuraciones de redes y túneles que le brindan comunicación.



Gestión de Almacenamiento: Permite soportar diferentes backends, pools y volúmenes de almacenamiento.



Gestión de Snapshot: Permite obtener y gestionar snapshots (instantáneas del contenedor en un determinado instante o estado).



Shell de configuración: Permite configurar por medio de línea de comandos.



Web de configuración: Permite configurar por medio una interfaz web.



Despliegue en la Nube: Permite desplegar servicios en la nube



Interacción: Permite realizar interacción entre contenedores y/o servicios.

Conceptos importantes



Imagen: Es una aplicación o conjunto de aplicaciones empaquetadas conjuntamente con todos los programas necesarios para su ejecución e interacción con el usuario, sus recursos y conectividad.



Contenedor: Es una instancia de una imagen, sobre la cual se pueden realizar una serie de cambios en su configuración y/o funcionalidad.

Requisitos

LXD para una mejor gestión de almacenamiento e interconexión de red, precisa los siguientes requisitos:

```
$ apt install zfsutils-linux
$ apt install bridge-utils
```

Instalación

Desde la versión Ubuntu 16.04, LXD viene instalado por defecto, sólo hace falta inicializarlo. En distribuciones como Debian y CentOS la instalación se la realiza por medio de **snap**.

Inicialización

Antes de poder hacer uso de todas y cada una de las bondades que representa el uso de LXD, es necesario inicializarla, aspecto que nos permite brindar los parámetros iniciales necesarios para la gestión correcta de contenedores respecto a su almacenamiento, manejo de recursos e interconectividad en la red, etc.

```
$ lxd init
Would you like to use LXD clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]: lxd
Name of the storage backend to use (btrfs, dir, lvm, zfs) [default=zfs]:
Create a new ZFS pool? (yes/no) [default=yes]:
Would you like to use an existing block device? (yes/no) [default=no]:
Size in GB of the new loop device (1GB minimum) [default=15GB]:
Would you like to connect to a MAAS server? (yes/no) [default=no]:
Would you like to create a new local network bridge? (yes/no) [default=yes]:
What should the new bridge be called? [default=lxdbr0]:
What IPv4 address should be used? (CIDR subnet notation, “auto” or “none”)
[default=auto]:
What IPv6 address should be used? (CIDR subnet notation, “auto” or “none”)
[default=auto]:
Would you like LXD to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]:
```

Gestión de repositorios e imágenes

Por defecto LXD provee 4 repositorios, donde se almacenan una diversidad de imágenes continuamente actualizadas y listas para ser utilizadas.

Listado de repositorios

Listado de repositorios remotos

$ lxc remote list

NAME	URL	PROTOCOL	AUTH TYPE	PUBLIC	STATIC
images	https://images.linuxcontainers.org	simplestreams		YES	NO
local (default)	unix://	lxd	tls	NO	YES
ubuntu	https://cloud-images.ubuntu.com/releases	simplestreams		YES	YES
ubuntu-daily	https://cloud-images.ubuntu.com/daily	simplestreams		YES	YES

Listado de un repositorio específico

$ lxc image list images:

ALIAS	FINGERPRINT	PUBLIC	DESCRIPTION	ARCH	SIZE	UPLOAD DATE
alpine/3.8 (3 more)	822fce21dd4e	yes	Alpine 3.8 amd64 (20180824_13:01)	x86_64	2.34MB	Aug 24, 2018 at 12:00am (UTC)
alpine/3.8/arm64 (1 more)	eeadf939ec2c	yes	Alpine 3.8 arm64 (20180824_13:03)	aarch64	2.23MB	Aug 24, 2018 at 12:00am (UTC)
centos/6 (3 more)	b3f86fe50a12	yes	Centos 6 amd64 (20180828_02:16)	x86_64	75.55MB	Aug 28, 2018 at 12:00am (UTC)
centos/6/i386 (1 more)	3c3f743f8826	yes	Centos 6 i386 (20180828_02:16)	i686	75.79MB	Aug 28, 2018 at 12:00am (UTC)
centos/7 (3 more)	798e34d8f6ba	yes	Centos 7 amd64 (20180828_02:16)	x86_64	83.45MB	Aug 28, 2018 at 12:00am (UTC)
debian/10 (7 more)	e8975959cfd0	yes	Debian buster amd64 (20180824_05:25)	x86_64	121.34MB	Aug 24, 2018 at 12:00am (UTC)
debian/10/arm64 (3 more)	ffa61dfbaab4	yes	Debian buster arm64 (20180824_05:25)	aarch64	116.28MB	Aug 24, 2018 at 12:00am (UTC)
debian/10/armel (3 more)	ff49d77edeea	yes	Debian buster armel (20180824_05:26)	armv7l	113.36MB	Aug 24, 2018 at 12:00am (UTC)
debian/10/armhf (3 more)	35e18f035964	yes	Debian buster armhf (20180824_05:26)	armv7l	113.36MB	Aug 24, 2018 at 12:00am (UTC)
debian/10/i386 (3 more)	1f2a2c752a4c	yes	Debian buster i386 (20180824_05:25)	i686	122.41MB	Aug 24, 2018 at 12:00am (UTC)

$ lxc image list ubuntu:

ALIAS	FINGERPRINT	PUBLIC	DESCRIPTION	ARCH	SIZE	UPLOAD DATE
a (5 more)	2d53824fdf89	yes	ubuntu 17.10 amd64 (release) (20180706)	x86_64	169.51MB	Jul 6, 2018 at 12:00am (UTC)
a/arm64 (2 more)	b80c64d35ad4	yes	ubuntu 17.10 arm64 (release) (20180706)	aarch64	153.62MB	Jul 6, 2018 at 12:00am (UTC)
a/armhf (2 more)	c4e73fb2c574	yes	ubuntu 17.10 armhf (release) (20180706)	armv7l	152.81MB	Jul 6, 2018 at 12:00am (UTC)
a/i386 (2 more)	e5885665a63a	yes	ubuntu 17.10 i386 (release) (20180706)	i686	172.43MB	Jul 6, 2018 at 12:00am (UTC)
a/ppc64el (2 more)	a6e28d423e1a	yes	ubuntu 17.10 ppc64el (release) (20180706)	ppc64le	176.11MB	Jul 6, 2018 at 12:00am (UTC)
a/s390x (2 more)	560a2959a07d	yes	ubuntu 17.10 s390x (release) (20180706)	s390x	161.08MB	Jul 6, 2018 at 12:00am (UTC)
b (9 more)	c76b5028c756	yes	ubuntu 18.04 LTS amd64 (release) (20180823)	x86_64	173.93MB	Aug 23, 2018 at 12:00am (UTC)
b/arm64 (4 more)	56abb1b22e19	yes	ubuntu 18.04 LTS arm64 (release) (20180823)	aarch64	158.11MB	Aug 23, 2018 at 12:00am (UTC)
b/armhf (4 more)	3078cced4cec	yes	ubuntu 18.04 LTS armhf (release) (20180823)	armv7l	157.41MB	Aug 23, 2018 at 12:00am (UTC)
b/i386 (4 more)	797df34e5632	yes	ubuntu 18.04 LTS i386 (release) (20180823)	i686	175.75MB	Aug 23, 2018 at 12:00am (UTC)
b/ppc64el (4 more)	3350732d633e	yes	ubuntu 18.04 LTS ppc64el (release) (20180823)	ppc64le	182.21MB	Aug 23, 2018 at 12:00am (UTC)
b/s390x (4 more)	d4b46d253b78	yes	ubuntu 18.04 LTS s390x (release) (20180823)	s390x	165.83MB	Aug 23, 2018 at 12:00am (UTC)

Listado de un repositorio por cierto valor de búsqueda

$ lxc image list images: 'centos'

ALIAS	FINGERPRINT	PUBLIC	DESCRIPTION	ARCH	SIZE	UPLOAD DATE
centos/6 (3 more)	b3f86fe50a12	yes	Centos 6 amd64 (20180828_02:16)	x86_64	75.55MB	Aug 28, 2018 at 12:00am (UTC)
centos/6/i386 (1 more)	3c3f743f8826	yes	Centos 6 i386 (20180828_02:16)	i686	75.79MB	Aug 28, 2018 at 12:00am (UTC)
centos/7 (3 more)	798e34d8f6ba	yes	Centos 7 amd64 (20180828_02:16)	x86_64	83.45MB	Aug 28, 2018 at 12:00am (UTC)

Listado de imágenes del repositorio local

$ lxc image list

ALIAS	FINGERPRINT	PUBLIC	DESCRIPTION	ARCH	SIZE	UPLOAD DATE
centos-7	5206a7bf4f53	no	Centos 7 amd64 (20180826_02:16)	x86_64	83.45MB	Aug 27, 2018 at 1:58am (UTC)
debian-stretch	263287f7adc4	no	Debian stretch amd64 (20180824_05:25)	x86_64	110.13MB	Aug 27, 2018 at 1:06am (UTC)

Listado por tipo de arquitectura

$ lxc image list amd64

ALIAS	FINGERPRINT	PUBLIC	DESCRIPTION	ARCH	SIZE	UPLOAD DATE
alpine/3.4 (3 more)	cc8b58012122	yes	Alpine 3.4 amd64 (20180627_17:50)	x86_64	2.04MB	Jun 27, 2018 at 12:00am (UTC)
alpine/3.5 (3 more)	9a99cef43379	yes	Alpine 3.5 amd64 (20180824_13:01)	x86_64	3.05MB	Aug 24, 2018 at 12:00am (UTC)
alpine/3.6 (3 more)	3bb9d07f30f4	yes	Alpine 3.6 amd64 (20180824_13:01)	x86_64	3.16MB	Aug 24, 2018 at 12:00am (UTC)
alpine/3.7 (3 more)	7a8f7eae58e9	yes	Alpine 3.7 amd64 (20180824_13:01)	x86_64	3.36MB	Aug 24, 2018 at 12:00am (UTC)
alpine/3.8 (3 more)	822fce21dd4e	yes	Alpine 3.8 amd64 (20180824_13:01)	x86_64	2.34MB	Aug 24, 2018 at 12:00am (UTC)
alpine/edge (3 more)	3170d6a85ed1	yes	Alpine edge amd64 (20180824_13:01)	x86_64	3.61MB	Aug 24, 2018 at 12:00am (UTC)
archlinux (5 more)	657e0d500203	yes	Archlinux current amd64 (20180828_01:27)	x86_64	135.37MB	Aug 28, 2018 at 12:00am (UTC)
centos/6 (3 more)	b3f86fe50a12	yes	Centos 6 amd64 (20180828_02:16)	x86_64	75.55MB	Aug 28, 2018 at 12:00am (UTC)
centos/7 (3 more)	798e34d8f6ba	yes	Centos 7 amd64 (20180828_02:16)	x86_64	83.45MB	Aug 28, 2018 at 12:00am (UTC)
debian/10 (7 more)	e8975959cfd0	yes	Debian buster amd64 (20180824_05:25)	x86_64	121.34MB	Aug 24, 2018 at 12:00am (UTC)

Información de una imagen

```
$ lxc image info ubuntu
Fingerprint: 5206a7bf4f5399e218e134e5b1798c11eec9f88a1aef35fdb8c696e2fa9ff1a4
Size: 83.45MB
Architecture: x86_64
Public: no
Timestamps:
    Created: 2018/08/26 00:00 UTC
    Uploaded: 2018/08/27 01:58 UTC
    Expires: never
    Last used: 2018/08/28 13:24 UTC
Properties:
    architecture: amd64
    description: Centos 7 amd64 (20180826_02:16)
    os: Centos
    release: 7
    serial: 20180826_02:16
Aliases:
    - centos-7
Cached: no
Auto update: disabled
Source:
    Server: https://images.linuxcontainers.org
    Protocol: simplestreams
    Alias: centos/7
```

Editar propiedades de una imagen

```
$ lxc image edit ubuntu
```

Eliminar una imagen

```
$ lxc image delete ubuntu
```

Exportar una imagen

```
$ lxc image export ubuntu .
```

Copiar una imagen al repositorio local

Copia sólo la imagen

```
$ lxc image copy ubuntu:14.04 local:
```

Copia la imagen con un alias

```
$ lxc image copy ubuntu:18.04 local: --alias ubuntu18
```

Copia la imagen con su alias original

```
$ lxc image copy ubuntu:18.04 local: --copy-aliases
```

Gestión de contenedores

Una vez que se tiene claro la ubicación de los repositorios y sus correspondientes imágenes, se puede proceder a gestionar y manipular los contenedores.

Creación

Crear y ejecutar un contenedor

```
$ lxc launch images:centos/7 webserver
Creating webserver
Starting webserver
```

Crear sin inicializar un contenedor

```
$ lxc init images:centos/7 webserver
```

Listado

Listado de contenedores

$ lxc list

```
+------------+---------+----------------------+-------------------------------------------+------------+-----------+
|    NAME    |  STATE  |         IPV4         |                   IPV6                    |    TYPE    | SNAPSHOTS |
+------------+---------+----------------------+-------------------------------------------+------------+-----------+
| mailserver | RUNNING | 10.37.113.145 (eth0) | fd42:354c:7b3f:8623:216:3eff:fe2f:7ea8 (eth0) | PERSISTENT | 0         |
+------------+---------+----------------------+-------------------------------------------+------------+-----------+
| webserver  | RUNNING | 10.37.113.122 (eth0) | fd42:354c:7b3f:8623:216:3eff:fe6c:6c53 (eth0) | PERSISTENT | 0         |
+------------+---------+----------------------+-------------------------------------------+------------+-----------+
```

Listado rápido

$ lxc list --fast

```
+------------+---------+--------------+----------------------+----------+------------+
|    NAME    |  STATE  | ARCHITECTURE |      CREATED AT      | PROFILES |    TYPE    |
+------------+---------+--------------+----------------------+----------+------------+
| mailserver | RUNNING | x86_64       | 2018/08/28 13:24 UTC | default  | PERSISTENT |
+------------+---------+--------------+----------------------+----------+------------+
| webserver  | RUNNING | x86_64       | 2018/08/28 13:22 UTC | default  | PERSISTENT |
+------------+---------+--------------+----------------------+----------+------------+
```

Información de un contenedor

```
$ lxc info webserver
Name: webserver
Remote: unix://
Architecture: x86_64
Created: 2018/08/28 13:22 UTC
Status: Running
Type: persistent
Profiles: default
Pid: 7110
Ips:
  eth0:      inet   10.37.113.122vethS9V0IO
  eth0:      inet6  fd42:354c:7b3f:8623:216:3eff:fe6c:6c53  vethS9V0IO
  eth0:      inet6  fe80::216:3eff:fe6c:6c53   vethS9V0IO
  lo:  inet   127.0.0.1
  lo:  inet6  ::1
Resources:
  Processes: 11
  Disk usage:
    root: 4.38MB
  CPU usage:
    CPU usage (in seconds): 2
  Memory usage:
    Memory (current): 24.53MB
    Memory (peak): 28.88MB
  Network usage:
    lo:
      Bytes received: 0B
      Bytes sent: 0B
      Packets received: 0
      Packets sent: 0
    eth0:
      Bytes received: 7.41kB
      Bytes sent: 2.29kB
      Packets received: 67
      Packets sent: 22
Snapshots:
  snap0 (taken at 2018/08/28 13:37 UTC) (stateless)
  snap-20180828 (taken at 2018/08/28 13:37 UTC) (stateless)
```

Operaciones con contenedores

Una vez creado el contenedor, éste puede ser tratado como un equipo independiente y sobre él mismo realizar las adecuaciones que sean necesarias acorde a las necesidades, como instalar paquetes, crear bases de datos, conectar a distintas redes, etc.

Iniciar contenedor

```
$ lxc start webserver
```

Detener contenedor

```
$ lxc stop webserver
```

Pausar un contenedor

```
$ lxc pause webserver
```

Eliminar un contenedor

```
$ lxc destroy webserver
```

Acceso a un contenedor

```
$ lxc exec webserver bash
```

Renombrar un contenedor

```
$ lxc move webserver webserver01
```

Clonar un contenedor

```
$ lxc copy webserver webserver10
```

Ejecución de comandos

La ejecución de comandos no necesariamente representa que primero debamos acceder al contenedor, sino que podemos realizarlo de forma remota sin acceder al mismo.

```
$ lxc exec webserver -- /bin/bash
$ lxc exec webserver -- apt-get update
$ lxc exec webserver -- ps -ef
```

Transporte de archivos

LXD permite transportar archivos desde y hacia los contenedores desde y hacia el HOST

Traer archivos

```
$ lxc file pull webserver/etc/hosts .
```

Llevar archivos

```
$ lxc file push hosts webserver/tmp/
```

Editar un archivo

```
$ lxc file edit webserver/etc/hosts
```

Próximamente

En los próximos artículos, mostraremos aspectos más avanzados de como gestionar y personalizar contenedores Linux.

Referencias

[1] http://www.linuxcontainers.org





Arduino

Aprendiendo Robótica III

Actualmente, el mundo atraviesa por una gran ola de cambios tecnológicos donde el uso de las tecnologías libres se está difundiendo más y más y llegando a límites nunca antes vistos, razón por la cual las personas deben adaptarse y aprender día a día de la realidad que están viviendo.

Una tecnología que ha copado la atención de grandes y pequeños en todos los ámbitos es la utilización de hardware libre, que permite crear entornos automatizados, prototipos, robótica educativa entre otros.

En nuestro anterior número, vimos el funcionamiento de los leds y también vimos la tremenda potencia que tiene el uso de las estructuras de repetitivas y las estructuras de control dentro un programa, ahora profundizaremos un poco más, veremos nuevos componentes y nuevos ejemplos.

No olvidemos nunca, ¡el único límite de nuestra imaginación somos nosotros, hagamos que ésta llegue hasta las estrellas!.

Y lo más importante, es tener ganas de aprender y de crecer dentro del mundo de la robótica!

Intentémoslo!

Interruptor

Alguna vez tuviste inquietud acerca del funcionamiento de un interruptor?, de como activa el funcionamiento de un foco dentro de nuestro hogar?, ó como funciona el timbre de nuestras casas?. Ahora descubriremos como funciona, podemos hacerlo de una manera muy simple.

Los interruptores tienen la función básica de establecer un puente de conectividad, esto quiere decir, que si los presionamos logramos que el puente de conectividad se establezca y por ende exista flujo de corriente que circule a través de ambos extremos y cuando no esté presionado el circuito interno permanece abierto y no existe flujo de corriente, dicho de otra forma no existe conectividad; los ejemplos mostrados en este número de la revista tratan de mostrarte su funcionamiento.



Botones una forma de simplificar las cosas

Les gustan los botones?, Por ejemplo, a mi me encanta presionar botones para que estos activen una función y simplifiquen las cosas.

A los botones y pulsadores se les puede encomendar activar o desactivar un conjunto de tareas, desde encender un led hasta cosas más complejas como realizar cálculos, envío de mensajes, etc.

Y si juntamos los botones con lo que ya sabemos de programación?, ¡Será algo mágico!, Por qué no lo hacemos?

Programando y divirtiéndonos a la vez, todo es posible; podemos hacer un sin fín de programas que hagan uso de uno o varios botones, y éstos al ser presionados activen una serie de tareas que les encomendaremos.

Encender un LED

A través de un botón podrás encender un led, y con otro podrás a pagarlo, para realizar este pequeño programa necesitas:

- 1 Protoboard
- 1 Arduino UNO
- 1 Led
- 3 Resistencias
- 2 botones
- Jumpers

Todos estos elementos conectados según el siguiente esquema:

```
int LED=2;
int BOTON01=8;
int BOTON02=9;

void setup()
{
  pinMode (LED, OUTPUT);
  pinMode (BOTON01, INPUT);
  pinMode (BOTON02, INPUT);
}
```

```
void loop()
{
  BOTON01 = digitalRead (8);

  if (BOTON01==HIGH){
    digitalWrite (LED, HIGH);
  }
  else
    {
    digitalWrite (LED, LOW);
    }

  BOTON02 = digitalRead (9);
  if (BOTON02==HIGH)
    {
    digitalWrite (LED, LOW);
    }
  else
    {
    digitalWrite (UNO, HIGH);
    }
}
```

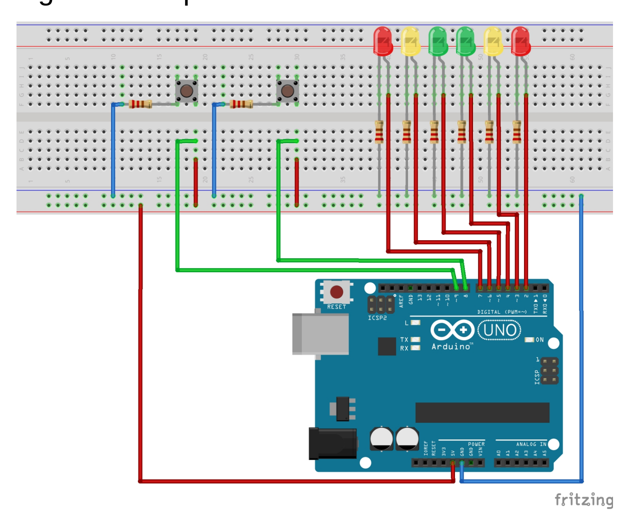
Sigamos aprendiendo! Ya vimos que prender un led a través de un botón no es muy complicado. Por qué no vamos poniéndole un poquito de dificultad a nuestro proyecto, recuerda que así podrás crecer y aprender mucho más. No te limites, recuerda que los programas que encuentras aquí solo son algunos ejemplos, y que tu puedes hacer muchos más, deja volar tu imaginación.

Secuencia rebote

Alguna vez escuchaste la secuencia de luz rebote?, bueno aquí te la enseño: es una secuencia de leds que se van encendiendo y apagando desde el centro hacia los extremos y viceversa; para nuestro ejemplo haremos que al presionar un botón la secuencia vaya desde el centro hacia los extremos y al presionar un segundo botón la secuencia vaya en sentido contrario cabe decir desde los extremos hacia el centro; ahora intentémoslo hacer, para realizar está secuencia necesitamos:

- Protoboard
- Arduino UNO
- 6 leds
- 2 Botones
- Jumpers
- 8 Resistencias

Todos estos elementos conectados según el siguiente esquema:




```
int UNO=2;
int DOS=3;
int TRES=4;
int CUATRO=5;
int CINCO=6;
int SEIS=7;
int t=1000;
int BOTON01=9;
int BOTON02=10
void setup ()
 {
  pinMode (UNO, OUTPUT);
  pinMode (DOS, OUTPUT);
  pinMode (TRES, OUTPUT);
  pinMode (CUATRO, OUTPUT);
  pinMode (CINCO, OUTPUT);
  pinMode (SEIS, OUTPUT);
  pinMode (BOTON01, INPUT);
  pinMode (BOTON02INPUT);
}

void loop()
{

BOTON01 = digitalRead (9);

if (BOTON01==HIGH)
  {
  digitalWrite (UNO, HIGH);
  digitalWrite (SEIS, HIGH);
  delay (t);
  digitalWrite (UNO, LOW);
  digitalWrite (SEIS, LOW);
  delay (t);
  digitalWrite (DOS, HIGH);
  digitalWrite (CINCO, HIGH);
  delay (t);
  digitalWrite (DOS, LOW);
  digitalWrite (CINCO, LOW);
  delay (t);
  digitalWrite (TRES, HIGH);
  digitalWrite (CUATRO, HIGH);
  delay (t);
  digitalWrite (TRES, LOW);
  digitalWrite (CUATRO, LOW);
  }
else
  {
  digitalWrite (UNO, LOW);
  digitalWrite (DOS, LOW);
  digitalWrite (TRES, LOW);
  digitalWrite (CUATRO, LOW);
  digitalWrite (CINCO, LOW);
  digitalWrite (SEIS, LOW);
  }
BOTON02 digitalRead (10);
if (BOTON02==HIGH)
  {
  digitalWrite (TRES, HIGH);
  digitalWrite (CUATRO, HIGH);
  delay (t);
  digitalWrite (TRES, LOW);
  digitalWrite (CUATRO, LOW);
  delay(t);
  digitalWrite (DOS, HIGH);
  digitalWrite (CINCO, HIGH);
  delay (t);
  digitalWrite (DOS, LOW);
  digitalWrite (CINCO, LOW);
  delay (t);
  digitalWrite (UNO, HIGH);
  digitalWrite (SEIS, HIGH);
  delay (t);
  digitalWrite (UNO, LOW);
  digitalWrite (SEIS, LOW);
else{
  digitalWrite (UNO, LOW);
  digitalWrite (DOS, LOW);
  digitalWrite (TRES, LOW);
  digitalWrite (CUATRO, LOW);
  digitalWrite (CINCO, LOW);
  digitalWrite (SEIS, LOW);
  }
}
```

Ahora veamos el mismo ejemplo pero con el uso de estructuras repetitivas y manejo de funciones.

```
int i;
int ADENTRO=1;
int AFUERA=0;

void setup()
{
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,INPUT);
  pinMode(9,INPUT);
}
```

```
void barrido (int sentido)
{
if (sentido==1)
  {
   for(i=7;i>=5;i--)
    {
     digitalWrite(i,HIGH);
     digitalWrite(8-i,HIGH);
     delay(100);
     digitalWrite(i,LOW);
     digitalWrite(8-i,LOW);
    }
  }
  else
   {
   for(i=5;i<=7;i++)
    {
     digitalWrite(i,HIGH);
     digitalWrite(8-i,HIGH);
     delay(100);
     digitalWrite(i,LOW);
     digitalWrite(8-i,LOW);
    }
   }
}

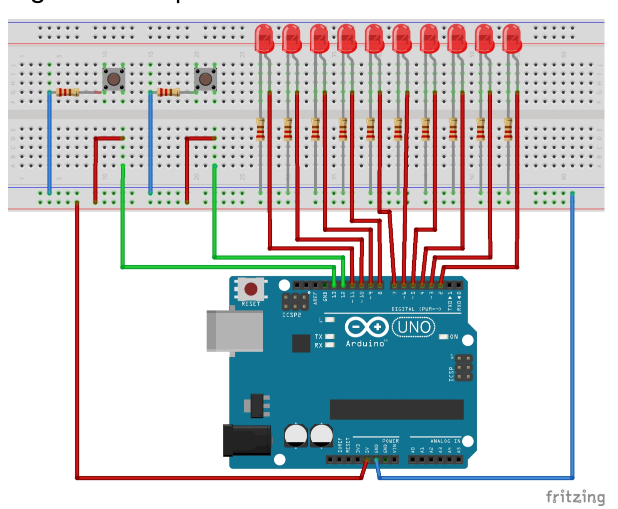
void loop()
{
  a= digitalRead(BOTON01);
  b= digitalRead(BOTON02);
  if (a==HIGH)
     {
     barrido(ADENTRO);
     a=LOW;
     }
  if (b==HIGH)
     {
     barrido(AFUERA);
     b=LOW;
     }
}
```

Secuencia efecto barrida

Alguna vez escuchaste el efecto barrida? Tal ves no, pero al hacer este programa, te darás cuenta de lo que esa frase significa. El siguiente programa esta en base a:

- 1 Protoboard
- 1 Arduino UNO
- 10 leds
- 2 Botones
- 12 Resitencias
- Jumpers

Todos estos elementos conectados según el siguiente esquema:




```
int a=2;
int b=3;
int c=4;
int d=5;
int e=6;
int f=7;
int g=8;
int h=9;
int i=10;
int j=11;
int x=1000;
int BOTON01=12;
int BOTON02=13;
void setup ()
 {
  pinMode (a, OUTPUT);
  pinMode (b, OUTPUT);
  pinMode (c, OUTPUT);
  pinMode (d, OUTPUT);
  pinMode (e, OUTPUT);
  pinMode (f, OUTPUT);
  pinMode (g, OUTPUT);
  pinMode (h, OUTPUT);
  pinMode (i, OUTPUT);
  pinMode (j, OUTPUT);
  pinMode (BOTON01, INPUT);
  pinMode (BOTON02, INPUT);
 }

void loop()
  {
  BOTON01 = digitalRead (12);
  if (BOTON01==HIGH){
  digitalWrite (a, HIGH);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
```

```
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (g, LOW);
digitalWrite (h, LOW);
digitalWrite (i, LOW);
digitalWrite (j, LOW);
delay (x);
digitalWrite (b, HIGH);
digitalWrite (a, LOW);
digitalWrite (c, LOW);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (g, LOW);
digitalWrite (h, LOW);
digitalWrite (i, LOW);
digitalWrite (j, LOW);
delay (x);
digitalWrite (c, HIGH);
digitalWrite (a, LOW);
digitalWrite (b, LOW);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (g, LOW);
digitalWrite (h, LOW);
digitalWrite (i, LOW);
digitalWrite (j, LOW);
delay (x);
digitalWrite (d, HIGH);
digitalWrite (g, HIGH);
digitalWrite (a, LOW);
digitalWrite (b, LOW);
digitalWrite (c, LOW);
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (h, LOW);
digitalWrite (i, LOW);
digitalWrite (j, LOW);
delay (x);
digitalWrite (e, HIGH);
digitalWrite (a, LOW);
digitalWrite (b, LOW);
digitalWrite (c, LOW);
digitalWrite (d, LOW);
digitalWrite (f, LOW);
digitalWrite (g, LOW);
digitalWrite (h, LOW);
digitalWrite (i, LOW);
digitalWrite (j, LOW);
delay (x);
digitalWrite (f, HIGH);
digitalWrite (a, LOW);
digitalWrite (b, LOW);
digitalWrite (c, LOW);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
digitalWrite (g, LOW);
digitalWrite (h, LOW);
digitalWrite (i, LOW);
digitalWrite (j, LOW);
delay(x);
digitalWrite (g, HIGH);
digitalWrite (a, LOW);
```

```
digitalWrite (b, LOW);
digitalWrite (c, LOW);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (h, LOW);
digitalWrite (i, LOW);
digitalWrite (j, LOW);
delay(x);
digitalWrite (h, HIGH);
digitalWrite (a, LOW);
digitalWrite (b, LOW);
digitalWrite (c, LOW);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (g, LOW);
digitalWrite (i, LOW);
digitalWrite (j, LOW);
delay(x);
digitalWrite (i, HIGH);
digitalWrite (a, LOW);
digitalWrite (b, LOW);
digitalWrite (c, LOW);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (g, LOW);
digitalWrite (h, LOW);
digitalWrite (j, LOW);
delay(x);
digitalWrite (j, HIGH);
digitalWrite (a, LOW);
digitalWrite (b, LOW);
digitalWrite (c, LOW);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (g, LOW);
digitalWrite (i, LOW);
digitalWrite (h, LOW);
delay(x);
else{
digitalWrite (a, LOW);
digitalWrite (b, LOW);
digitalWrite (c, LOW);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (g, LOW);
digitalWrite (h, LOW);
digitalWrite (i, LOW);
digitalWrite (j, LOW);
}
}
BOTON02 = digitalRead (13);
if (BOTON02==HIGH){
// Segunda vuelta
digitalWrite (j, HIGH);
digitalWrite (a, LOW);
digitalWrite (b, LOW);
digitalWrite (c, LOW);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
```

```
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
  digitalWrite (i, LOW);
  digitalWrite (h, LOW);
  delay(x);
  digitalWrite (i, HIGH);
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
  digitalWrite (h, LOW);
  digitalWrite (j, LOW);
  delay(x);
  digitalWrite (h, HIGH);
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
  digitalWrite (i, LOW);
  digitalWrite (j, LOW);
  delay(x);
  digitalWrite (g, HIGH);
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (h, LOW);
  digitalWrite (i, LOW);
  digitalWrite (j, LOW);
  delay(x);
  digitalWrite (f, HIGH);
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, LOW);
  digitalWrite (g, LOW);
  digitalWrite (h, LOW);
  digitalWrite (i, LOW);
  digitalWrite (j, LOW);
  delay(x);
  digitalWrite (e, HIGH);
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
  digitalWrite (h, LOW);
  digitalWrite (i, LOW);
  digitalWrite (j, LOW);
  delay (X);
  digitalWrite (d, HIGH);
  digitalWrite (g, HIGH);
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (h, LOW);
  digitalWrite (i, LOW);
  digitalWrite (j, LOW);
  delay (x);
  digitalWrite (c, HIGH);
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
  digitalWrite (h, LOW);
  digitalWrite (i, LOW);
  digitalWrite (j, LOW);
  delay (x);
  digitalWrite (b, HIGH);
  digitalWrite (a, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
  digitalWrite (h, LOW);
  digitalWrite (i, LOW);
  digitalWrite (j, LOW);
  delay (x);
  digitalWrite (a, HIGH);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
  digitalWrite (h, LOW);
  digitalWrite (i, LOW);
  digitalWrite (j, LOW);
  delay (x);
  }
  else
  {
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
  digitalWrite (h, LOW);
  digitalWrite (i, LOW);
  digitalWrite (j, LOW);
  }
  }
}
```

Ahora mostramos el mismo ejemplo pero con el uso de estructuras repetitivas y el uso de funciones.

```
int i;
int IZQUIERDA=1;
int DERECHA=0;

void setup()
{
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
  pinMode(12,INPUT);
  pinMode(13,INPUT);
}

void barrido (int sentido)
{
if sentido==1
  {
   for(i=11;i>=2;i--)
    {
     digitalWrite(i,HIGH);
     delay(100);
     digitalWrite(i,LOW);
    }
  }
  else
   {
   for(i=2;i<=11;i++)
    {
     digitalWrite(i,HIGH);
     delay(100);
     digitalWrite(i,LOW);
    }
   }
}


void loop()
{

  a= digitalRead(BOTON01);
  b= digitalRead(BOTON02);

  if (a==HIGH)
     {
     barrido(IZQUIERDA);
     a=LOW;
     }
  if (b==HIGH)
     {
     barrido(DERECHA);
     b=LOW;
     }
}
```

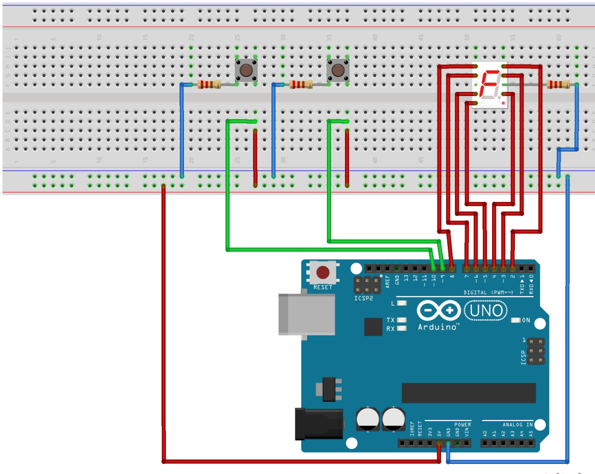
Contador ascendente y descendente

Ahora veremos como reutilizar el programa del manejo de un display de 7 segmentos, realizado en el anterior número de la revista; con un botón haremos que un contador ascienda de forma automática y con el segundo botón el contador descienda de forma automática.

Para ésto necesitaremos

- 1 Protoboard
- 1 Arduino UNO
- 1 display de 7 segmentos
- 3 Resistencias
- 2 botones
- jumpers

Todos estos elementos conectados según el siguiente esquema:




```
int pausa=1000;
int ASCIENDE=1;
int DESCIENDE=0;
void setup()
{
// Asignación de las salidas digitales
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9,INPUT);
  pinMode(10,INPUT);
}
```

Robótica

```
//Función que enciende o apaga los
segmentos según corresponda
void     display (int a, int b, int c,
int d,
int e, int f, int g)
{
//Se reciben 7 variables asignadas a
cada segmento
  digitalWrite (2,a);
  digitalWrite (3,b);
  digitalWrite (4,c);
  digitalWrite (5,d);
  digitalWrite (6,e);
  digitalWrite (7,f);
  digitalWrite (8,g);
}

void contador (int sentido)
{
  if sentido==1
  {
    display (1,1,1,1,1,1,0);
    delay(pausa);
    display (0,1,1,0,0,0,0);
    delay(pausa);
    display (1,1,0,1,1,0,1);
    delay(pausa);
    display (1,1,1,1,0,0,1);
    delay(pausa);
    display (0,1,1,0,0,1,1);
    delay(pausa);
    display (1,0,1,1,0,1,1);
    delay(pausa);
    display (1,0,1,1,1,1,1);
    delay(pausa);
    display (1,1,1,0,0,0,0);
    delay(pausa);
    display (1,1,1,1,1,1,1);
    delay(pausa);
    display (1,1,1,0,0,1,1);
    delay(pausa);
  }
  else
  {
    display (1,1,1,0,0,1,1);
    delay(pausa);
    display (1,1,1,1,1,1,1);
    delay(pausa);
    display (1,1,1,0,0,0,0);
    delay(pausa);
    display (1,0,1,1,1,1,1);
    delay(pausa);
    display (1,0,1,1,0,1,1);
    delay(pausa);
    display (0,1,1,0,0,1,1);
    delay(pausa);
    display (1,1,1,1,0,0,1);
    delay(pausa);
    display (1,1,0,1,1,0,1);
    delay(pausa);
    display (0,1,1,0,0,0,0);
    delay(pausa);
    display (1,1,1,1,1,1,0);
    delay(pausa);
  }
}

void loop()
{
  a= digitalRead(BOTON01);
  b= digitalRead(BOTON02);

  if (a==HIGH)
     {
     contador(ASCIENDE);
     a=LOW;
     }
  if (b==HIGH)
     {
     contador(DESCIENDE);
     b=LOW;
     }
}
```

Es algo mágico trabajar con botones, este es uno más de los muchos elementos que Arduino te permite utilizar, te gusta? Sigue intentando más proyectos, verás que aprendes más y que tu crecimiento en Arduino se hace cada vez mayor.

Referencias

[1] http://www.arduino.cc





GPLI
Gestión de Servicios TI

La gestión de servicios TI (ITSM – IT Service Management) es la forma de aportar valor al negocio mediante soluciones de TI mediante la combinación de procesos, personas y tecnología. Una herramienta para la gestión de servicios de TI es el GLPi (en francés, Gestionnaire Libre de Parc Informatique) mediante una solución bastante robusta.

Introducción

GLPi es una herramienta ITSM y una aplicación web libre y de código abierto para la gestión de los sistemas de información, para el manejo y control de los cambios en tu infraestructura informática de manera sencilla, resolver problemas emergentes (Service Desk) de manera eficiente y además hace posible el control fiable sobre el presupuesto y gastos que realiza tu compañía en IT.



Entre las características del GLPi están:

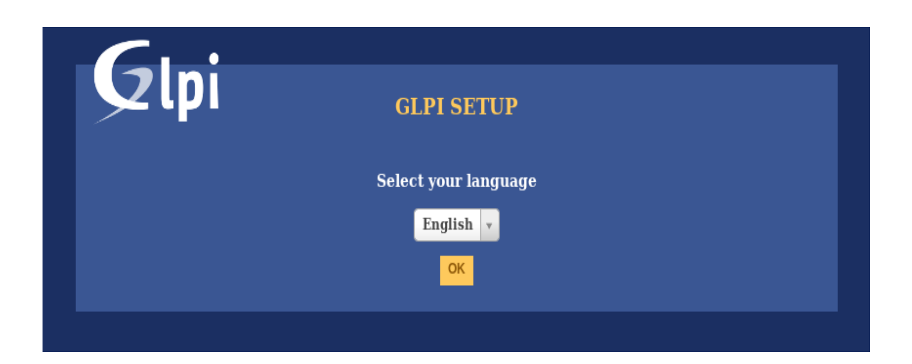
- Administración de grandes infraestructuras de TI.
- Seguimiento del ciclo de vida de los activos de TI.
- Compatible con ITIL.
- Inventario de activos de TI.
- Uso de diferentes tipos de Plugins.



Instalación y Ejecución

Como requisito debemos tener instalado Apache 2, PHP 9.2 y MySQL 5.6.

Podemos bajar la versión de GLPi de **`https://glpi-project.org/downloads/`** o de **`https://github.com/glpi-project/glpi/releases`**, se bajará el archivo **`glpi-9.3.tgz`** el cual se lo debe descomprimir dentro las carpetas de Apache e ingresar a la URL main address de Apache (ejemplo **`https://172.0.0.1`**) y se abrirá la siguiente página:



Luego de escoger el idioma, seleccionamos **`OK`** y se abrirá la siguiente ventana donde se tiene los términos y condiciones para el uso de este software:



A continuación, se abrirá la siguiente pantalla para elegir la opción de instalación nueva o de upgrade, seleccionamos **`INSTALL`** y se correrá la validación de los prerequisitos:

GLPI SETUP

Step 0

Checking of the compatibility of your environment with the execution of GLPI

Test done	Results
Testing PHP Parser	✔
Sessions test	✔
Test if Session_use_trans_sid is used	✔
mysqli extension test	✔
ctype extension test	✔
fileinfo extension test	✔
json extension test	✔
mbstring extension test	✔
zlib extension test	✔
curl extension test	✔
gd extension test	✔
simplexml extension test	✔
xml extension test	✔
imap extension test	✔
APCu extension test	✔
xmlrpc extension test	✔
ldap extension test	ldap extension is not present
Zend OPcache extension test	Zend OPcache extension is not present
Allocated memory test	✔
Checking write permissions for setting files	✔
Checking write permissions for document files	✔
Checking write permissions for dump files	✔
Checking write permissions for session files	✔
Checking write permissions for automatic actions files	✔
Checking write permissions for graphic files	✔
Checking write permissions for lock files	✔
Checking write permissions for plugins document files	✔
Checking write permissions for temporary files	✔
Checking write permissions for cache files	✔
Checking write permissions for rss files	✔
Checking write permissions for upload files	✔
Checking write permissions for pictures files	✔
Checking write permissions for log files	✔
SELinux mode is Enforcing	✔
SELinux boolean configuration for httpd_can_network_connect --> on	✔
SELinux boolean configuration for httpd_can_network_connect_db --> on	✔
SELinux boolean configuration for httpd_can_sendmail --> on	✔

Do you want to continue?

Continue | Try again

Si cumplimos con todos los requisitos, debemos elegir **CONTINUE** y se abrirá la ventana para parametrizar la conexión a la base de datos:



Después de llenar los tres campos de forma apropiada, escogemos **CONTINUE** y se abrirá la siguiente ventana, donde aparecen las bases de datos encontradas con las credenciales que se llenaron en la ventana anterior:



Escogemos la base de datos que utilizaremos para el **GLPi** y se inicializará la base de datos con todo lo necesario:



Después de configurar la base de datos se debe seleccionar **CONTINUE** y se abrirá la siguiente ventana, donde se pide enviar información de telemetría para poder realizar futuras mejoras, es opcional el seleccionar el poder enviar información estadística:



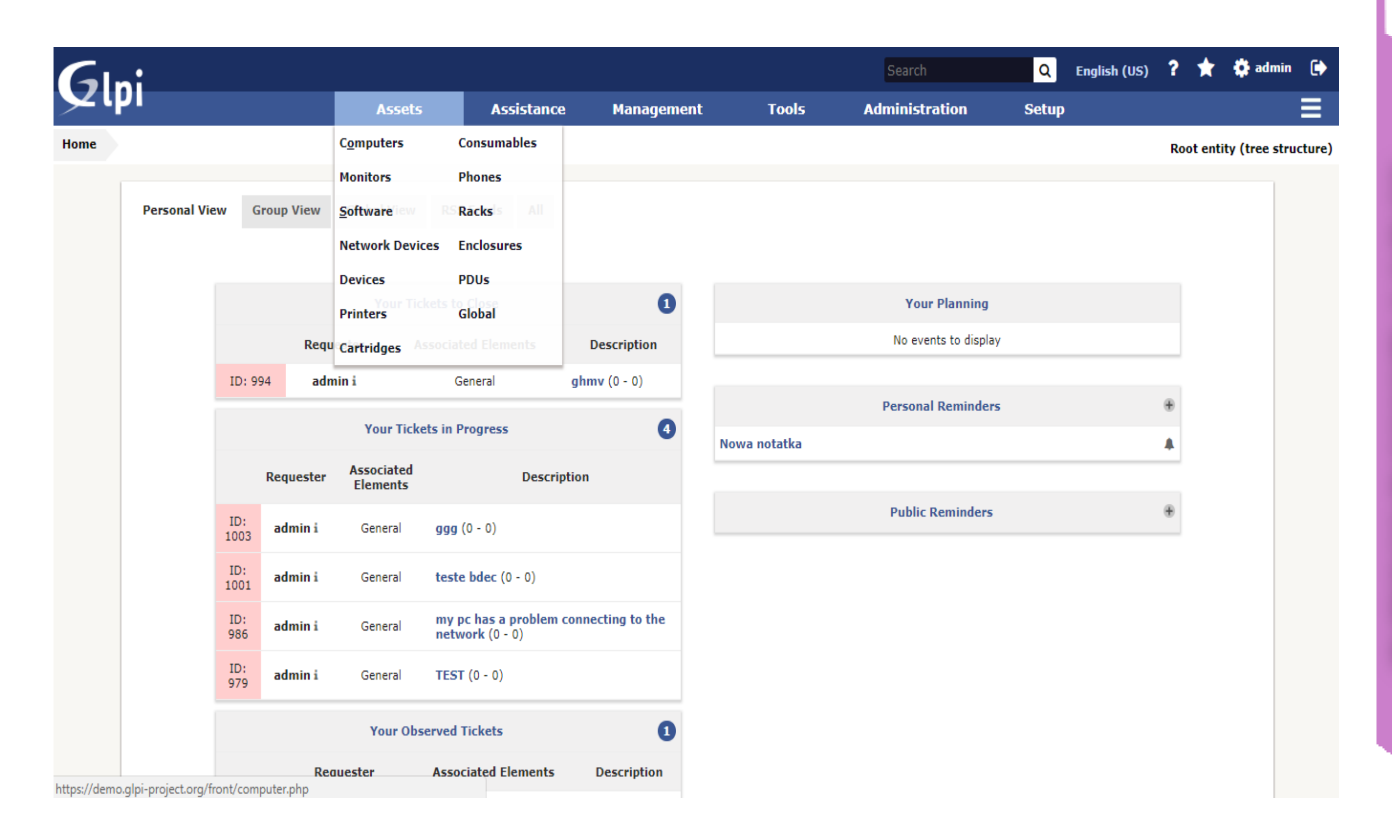
Por último, se desplegará la ventana de finalización de la instalación:



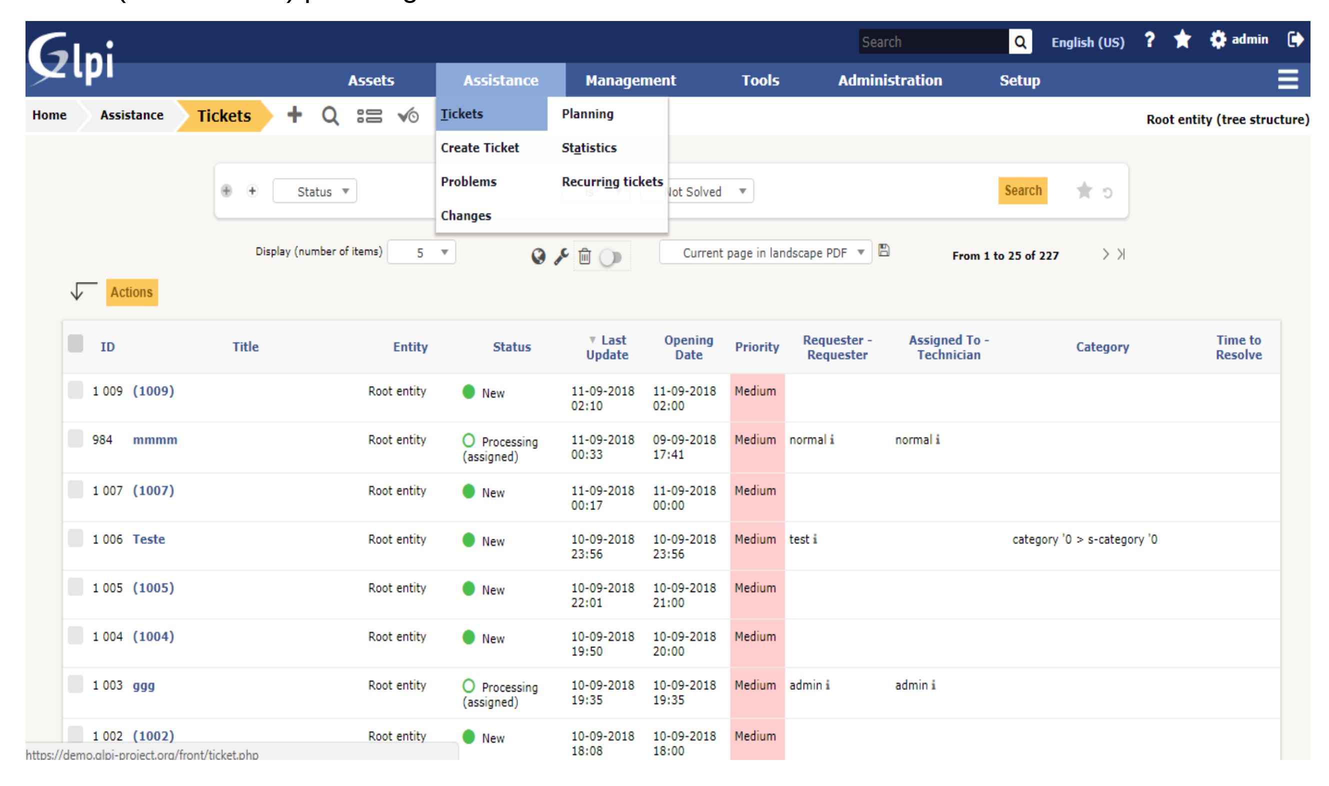
Para el acceso al **GLPi** se crearon los siguientes usuarios/password por defecto:

- Cuenta administradora: **glpi/glpi**
- Cuenta técnica: **tech/tech**
- Cuenta de usuario normal: **normal/normal**

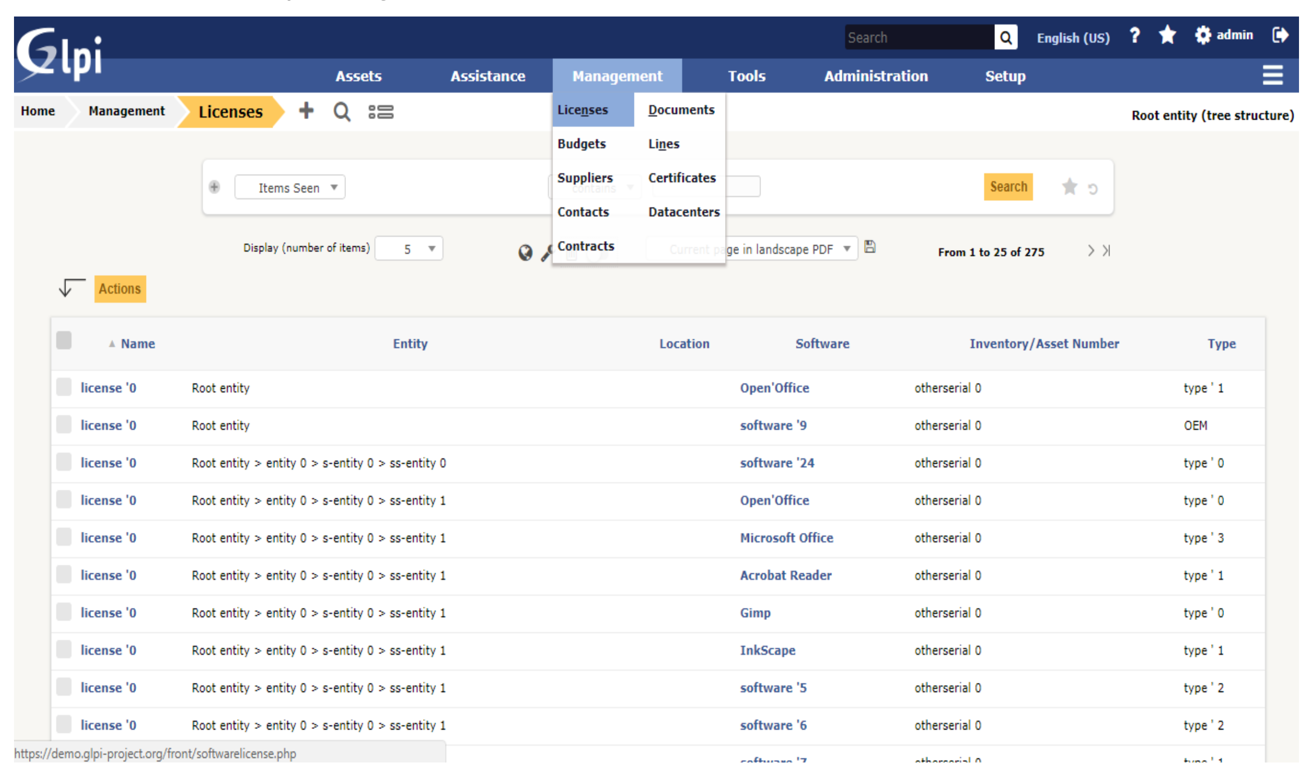
Una vez logueado como administrador se obtiene la siguiente pantalla:



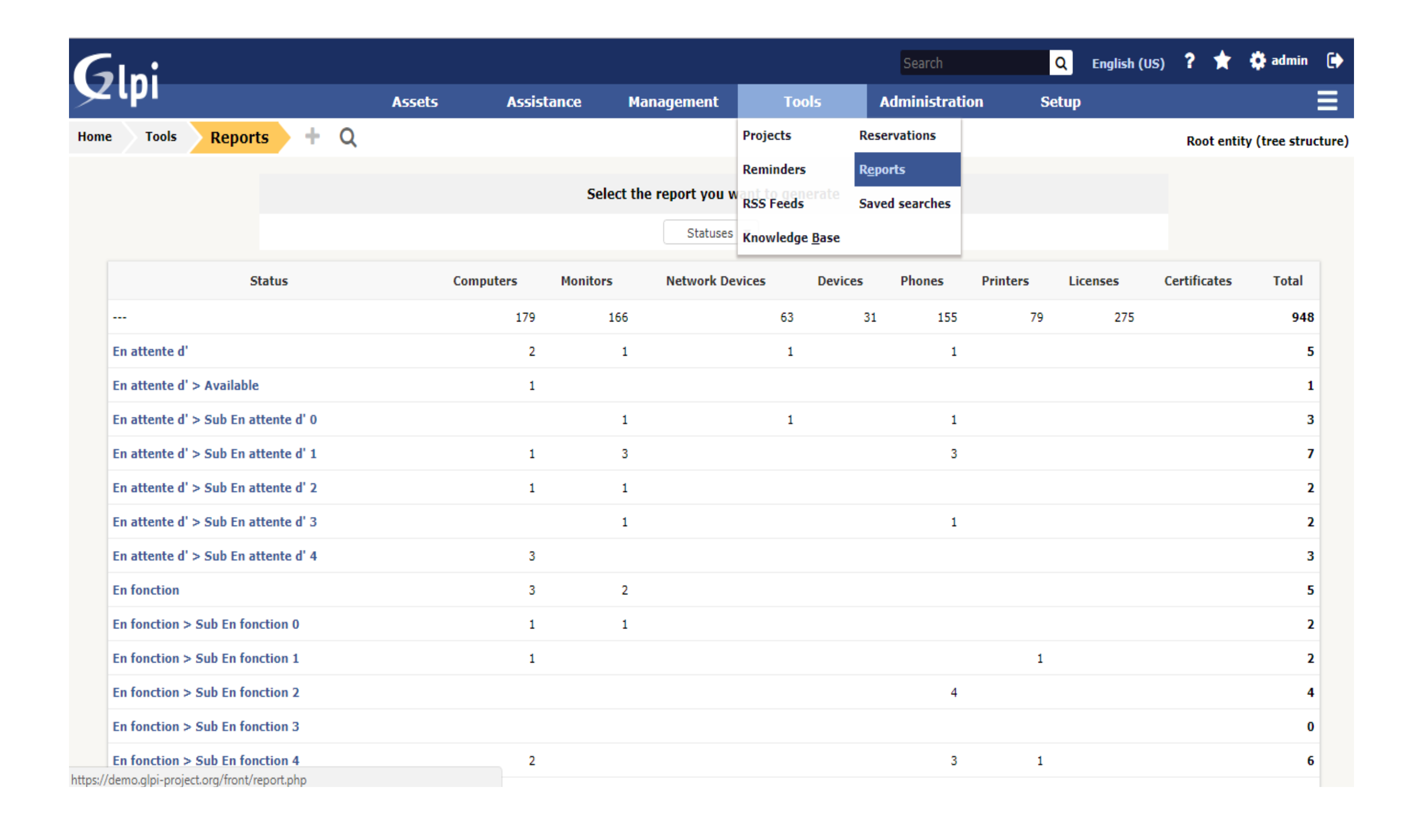
En este ejemplo ya se crearon varios activos de TI (**ASSETS**) y se tiene configurado un Service Desk (**ASSISTANCE**) para la gestión de Tickets de Service Desk:

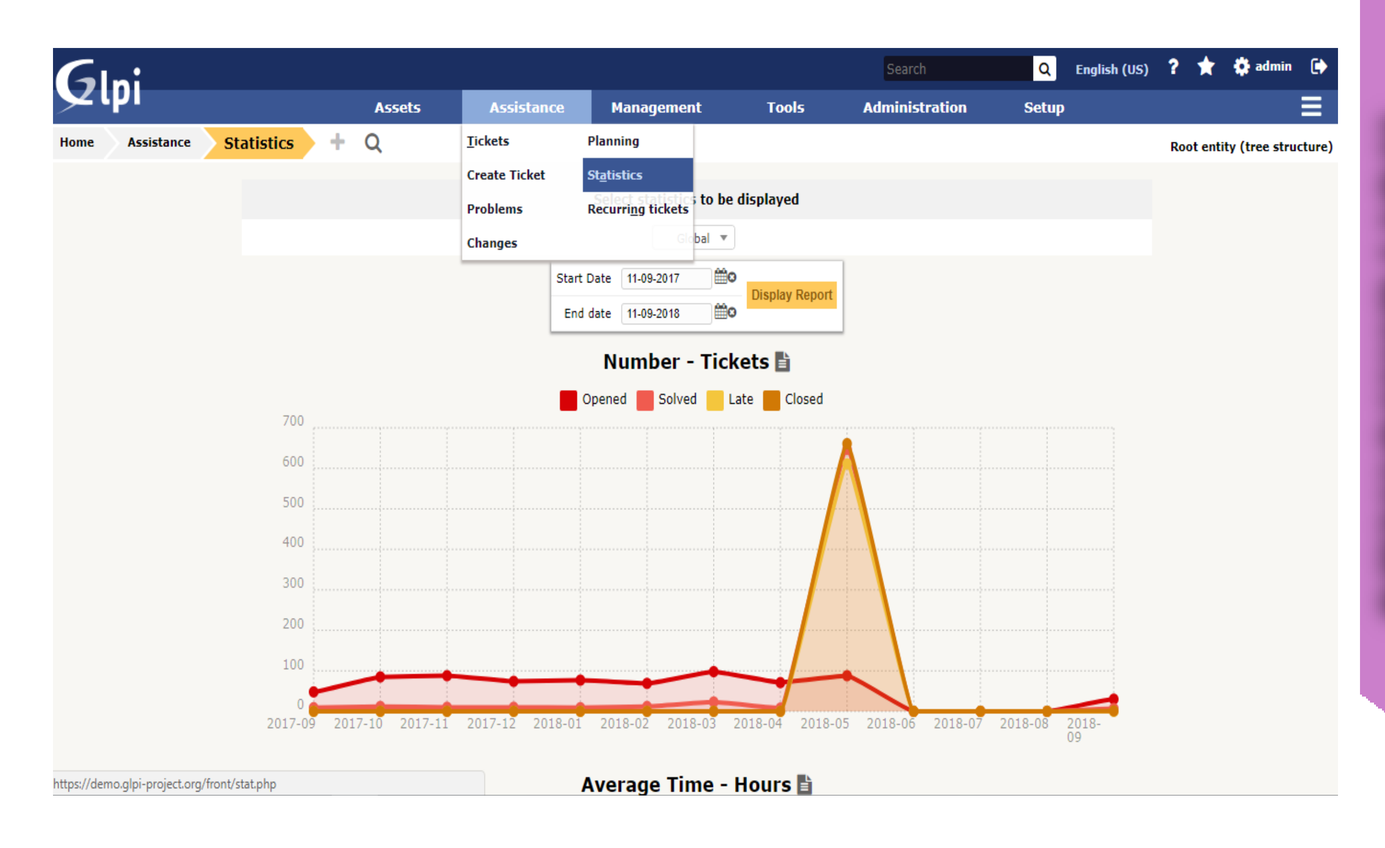


Adicionalmente, se puede gestionar las licencias de software:



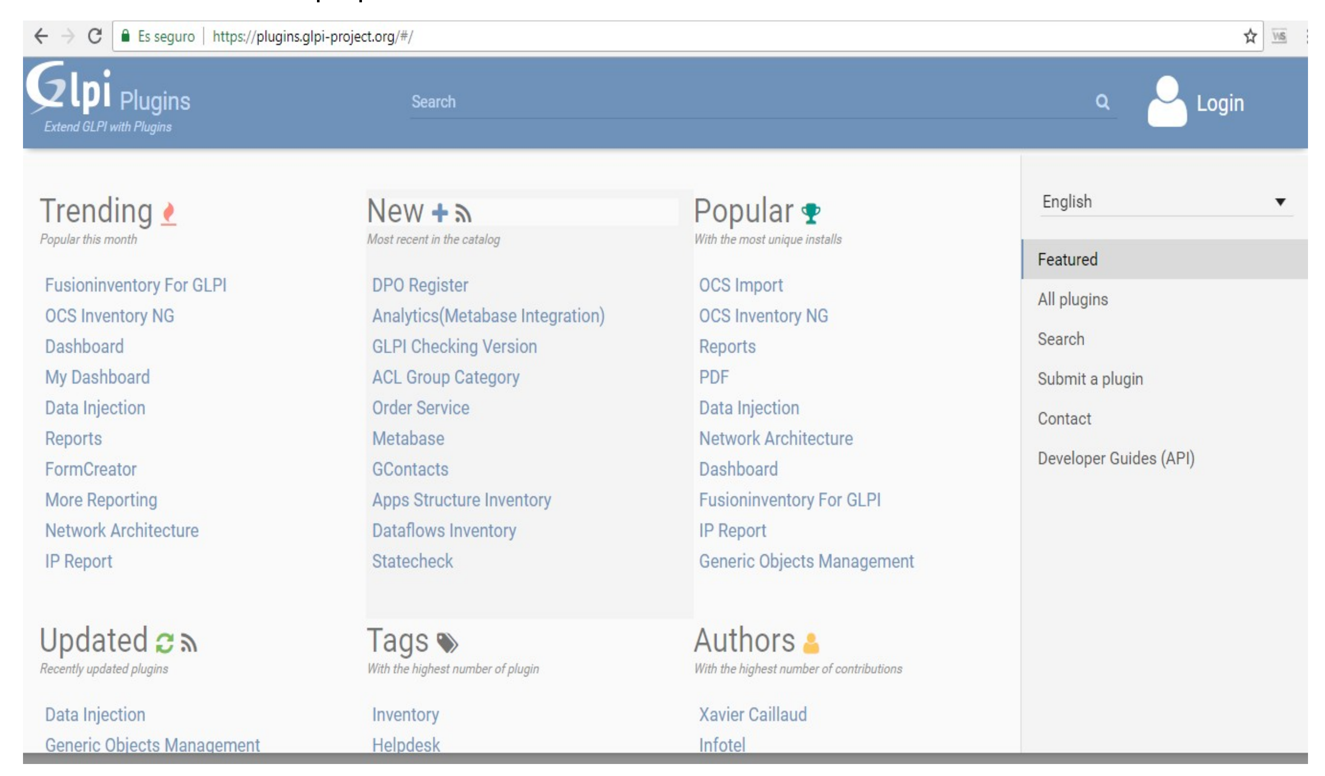
Con el GLPI se pueden gestionar todos las partes que conforman los Servicios de TI. Adicionalmente, se cuentan con reportes de "**Activos de TI**" y "**Tickes de Service Desk**", como por ejemplo:





Plugins

El GLPi se puede integrar con el Software **OCS** (Inventario de Activos TI) y se pueden instalar **PLUGINS** con distintos propósitos:



Conclusiones

- Este tipo de herramienta ITSM, son de mucha ayuda para la gestión de servicios TI.
- La gran variedad de Plugins disponibles para GLPi, aportan de gran manera para mejorar la gestión de servicios TI.

Referencias

[1] https://glpi-project.org

[2] https://plugins.glpi-project.org/





Desarrollo

Actualmente en el desarrollo moderno de aplicaciones web se utilizan distintos Frameworks que son herramientas que nos dan un esquema de trabajo y una serie de utilidades y funciones que nos facilita y nos abstrae de la construcción de aplicaciones web dinámicas.

Flask es un framework minimalista escrito en Python, concebido para facilitar el desarrollo de Aplicaciones Web bajo el patrón MVC, de forma rápida y con un mínimo número de líneas de código

En el anterior número de la revista creamos **geistesblitze**, una aplicación Web para registrar ideas con Flask. La aplicación carece de operaciones para poder actualizar o borrar las ideas y es (a propósito) lo más simple posible, mientras muestra lo fácil que es crear una aplicación Web en Flask usando un base de datos a través de SQLAlchemy.

Queremos implementar una interfaz más moderna, que será el "*Frontend*", de una aplicación de una sola página en Javascript que consumirá los servicios REST del "*Backend*" que implementará la aplicación Web en Flask.

Para eso, antes tenemos que convertir las vistas de nuestra aplicación en servicios REST que acepten y respondan usando JSON. Además de crear una autorización compatible con el uso de los servicios independientemente de una página Web y de acuerdo a los principios que definen a una interfaz REST.

En una próxima edición cerraremos la serie, creando la interfaz usando jQuery.

geistesblitze v2

El primer paso es deshacernos de las plantillas y las vistas que generan páginas Web en base a ellas, y crear una interfaz **REST**, que ofrezca los servicios y responda usando **JSON**, para:

- registrar usuarios
- añadir ideas
- listar todas las ideas
- listar una idea
- actualizar una idea
- eliminar una idea
- obtener un token.

El desarrollo de la aplicación se puede seguir en GitHub, en la rama v2 [1]

Transferencia de Estado Representacional – REST

Transferencia de estado representacional, o REST por su sigla en ingles, es un estilo de arquitectura de software para sistemas Web. Sus orígenes se remontan a la tesis de doctorado de Roy Fielding en el año 2000, y se ha convertido en los últimos años en una especie de lengua franca entre desarrolladores.

Actualmente REST se usa en un sentido más amplio al referido en la tesis de Fielding para describir cualquier interfaz que utilice directamente HTTP y sus métodos, para obtener datos o indicar la ejecución de operaciones sobre los datos en cualquier formato, ya sea XML o JSON.

Los aspectos más importantes de una interfaz REST son:

- es un protocolo en que cada mensaje HTTP contiene toda la información necesaria,
- se hace uso de los diferentes métodos POST, GET, PUT y DELETE de HTTP para crear, leer, actualizar o borrar, respectivamente los recursos.
- cada recurso es direccionable únicamente a través de su identificador de recursos uniforme, o URI (*uniform resource identifier*).

Servicios

Los servicios se comunicarán con el Frontend usando JSON. Recibirán requests en JSON y responderán también en JSON, es decir los parámetros son elementos de un objeto JSON.

Registrar usuarios

La vista para registrar a nuevos usuarios utiliza el método post de HTTP y acepta el nombre del usuario y su contraseña como elementos de un objeto JSON. Si no se proporciona el nombre de usuario o la contraseña, se devuelve un error 400 "*Bad Request*" y si el usuario ya existe se devuelve un error 409 "*Conflict*".

Luego de crear un nuevo usuario, se devuelve en id y el nombre de usuario en un objeto JSON con el estatus 201 "*Created*".

```
@app.route('/api/users', methods=['POST'])
def register_user():
    username = request.json.get('username')
    password = request.json.get('password')

    if username is None or password is None:
        abort(400)

    if User.query.filter_by(username=username).first() is not None:
        abort(409)

    user = User(username=username, password=password)

    db.session.add(user)
    db.session.commit()

    return jsonify(dict(id=user.id, username=user.username)), 201
```

La clave no es almacenada directamente, sino a través de un "*hash*". En la clase User hay métodos para generar un hash en base a la contraseña que es almacenado en lugar de la contraseña original (texto plano), verificar si la contraseña es correcta, es decir si el hash de la contraseña corresponde al almacenado y para generar un token.



Verificar la existencia de un usuario

Para poder verificar que el usuario ha sido creado, podemos usar un servicio que no requiere ningún tipo de autorización adicional, utilizando el método GET de HTTP, que acepta el ID del usuario como parámetro parte del URL y nos devuelve el id y el nombre de el usuario.

```
@app.route('/api/users/<int:user_id>')
def get_user(user_id):
    user = User.query.get(user_id)

    if not user:
        abort(400)

    return jsonify(dict(id=user.id, username=user.username))
```

Ahora que tenemos la posibilidad de registrar nuevas y nuevos usuarios, vamos a necesitar una función para verificar que la contraseña es correcta y almacenar al usuario en el objeto global g.

```
@auth.verify_password
def verify_password(username_or_token, password):
    user = User.verify_auth_token(username_or_token)

    if not user:
        user = User.query.filter_by(username=username_or_token).first()
        if not user or not user.verify_password(password):
            return False

    g.user = user

    return True
```

Generar un token

Para poder acceder al resto de la interfaz REST vamos a usar un token, que será utilizado para crear, cambiar, listar y borrar las ideas que vayamos a tener.

```
@app.route('/api/token')
@auth.login_required
def get_auth_token():
    token = g.user.generate_auth_token()

    return jsonify(dict(token=token.decode('ascii')))
```

Añadir una idea

Para añadir una idea vamos a utilizar el método post de HTTP. Acepta el nombre y la descripción de la idea como elementos JSON. La respuesta serán simplemente el ID, el nombre y la descripción de la idea que acabamos de crear y el estatus 201 “*Created*”.

```
@app.route('/api/ideas/', methods=['POST'])
@auth.login_required
def add_idea():
    """adds a new idea."""
    name = request.json.get('name')
    description = request.json.get('description')

    idea = Idea(name=name, description=description)
    idea.user = g.user

    db.session.add(idea)
    db.session.commit()
```

```
    return jsonify(dict(id=idea.id,
                        name=idea.name,
                        description=idea.description)), 201
```

Actualizar una idea

Para actualizar una idea, es decir cambiar el nombre o la descripción de la misma, vamos a utilizar el método PUT de HTTP. Acepta el nombre y la descripción de la idea como elementos JSON. Si la idea que queremos cambiar no existe, devolveremos el error 404 "*Not Found*", y si la idea que queremos cambiar le pertenece a otro usuario, devolveremos el error 403 "*Forbidden*".

Una vez cambiada la idea, devolveremos el id, el nombre y la descripción de la idea con el estatus 201, "*Created*".

```
@app.route('/api/ideas/<int:idea_id>', methods=['PUT'])
@auth.login_required
def update_idea(idea_id):
    idea = Idea.query.filter_by(id=idea_id).first()

    name = request.json.get('name')
    description = request.json.get('description')

    if not idea:
        abort(404)

    if idea.user_id != g.user.id:
        abort(403)

    idea.name = name
    idea.description = description

    db.session.add(idea)
    db.session.commit()

    return jsonify(dict(id=idea.id,
                        name=idea.name,
                        description=idea.description)), 201
```

Listar las ideas

Para obtener la lista de ideas utilizaremos el método GET de HTTP y devolveremos una lista con todas las idea propias.

```
@app.route('/api/ideas/')
@auth.login_required
def get_ideas():
    """"gets all the ideas of the current user."""
    ideas = [dict(id=idea.id, name=idea.name, description=idea.description)
             for idea in Idea.query.filter_by(user=g.user).all()]

    return jsonify(ideas)
```

Obtener los detalles de una idea

Para obtener los detalles de una idea, tenemos que verificar que la idea exista y sea del usuario, en caso contrario devolveremos el error 404 "*Not found*" o 403 "*Forbidden*".

```
@app.route('/api/ideas/<int:idea_id>')
@auth.login_required
def get_idea(idea_id):
    idea = Idea.query.filter_by(id=idea_id).first()
```

```
    if not idea:
        abort(404)

    if idea.user_id != g.user.id:
        abort(403)

    return jsonify(dict(id=idea.id,
                        name=idea.name,
                        description=idea.description))
```

Eliminar una idea

Para eliminar una idea utilizaremos el método DELETE de HTTP. En caso de que la idea no exista, devolveremos el error 404 “*Not Found*”, y si la idea no es del usuario, devolveremos el error 403 “*Forbidden*”.

Una vez eliminada la idea responderemos con el código de estatus 204, “*No Content*” y sin datos adicionales.

```
@app.route('/api/ideas/<int:idea_id>', methods=['DELETE'])
@auth.login_required
def delete_idea(idea_id):
    idea = Idea.query.filter_by(id=idea_id).first()

    if not idea:
        abort(404)

    if idea.user_id != g.user.id:
        abort(403)

    db.session.delete(idea)
    db.session.commit()

    return "", 204
```

Los servicios en acción

Para probar que nuestra aplicación funciona podemos utilizar por ejemplo podemos usar httpie[2].

```
(venv) $ pip install httpie
```

Luego de arrancar la aplicación en una terminal usando el entorno virtual, podemos probar los servicios desde otra terminal.

```
~/Projects/geistesblitze
~/Projects/geistesblitze    ~/Projects/geistesblitze
(venv) ~/Projects/geistesblitze$ export FLASK_APP=geistesblitze
(venv) ~/Projects/geistesblitze$ flask run
 * Serving Flask app "geistesblitze"
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Registrar usuarios

```
~/Projects/geistesblitze
~/Projects/geistesblitze ×    ~/Projects/geistesblitze ×
(venv) ~/Projects/geistesblitze$ http --json POST :5000/api/users username=ers password=secreto
HTTP/1.0 201 CREATED
Content-Length: 26
Content-Type: application/json
Date: Tue, 11 Sep 2018 01:24:36 GMT
Server: Werkzeug/0.14.1 Python/3.6.5

{
    "id": 1,
    "username": "ers"
}

(venv) ~/Projects/geistesblitze$
```

Verificar la existencia de un usuario

```
~/Projects/geistesblitze
~/Projects/geistesblitze ×    ~/Projects/geistesblitze ×
(venv) ~/Projects/geistesblitze$ http --json GET :5000/api/users/1
HTTP/1.0 200 OK
Content-Length: 26
Content-Type: application/json
Date: Tue, 11 Sep 2018 01:26:14 GMT
Server: Werkzeug/0.14.1 Python/3.6.5

{
    "id": 1,
    "username": "ers"
}

(venv) ~/Projects/geistesblitze$
```

Generar un token

```
~/Projects/geistesblitze
~/Projects/geistesblitze ×    ~/Projects/geistesblitze ×
(venv) ~/Projects/geistesblitze$ http --json --auth ers:secreto GET :5000/api/token
HTTP/1.0 200 OK
Content-Length: 135
Content-Type: application/json
Date: Tue, 11 Sep 2018 01:26:57 GMT
Server: Werkzeug/0.14.1 Python/3.6.5

{
    "token": "eyJhbGciOiJIUzI1NiIsImlhdCI6MTUzNjYyOTIxNywiZXhwIjoxNTM2NjI5ODE3fQ.eyJpZCI6MX0.5E4CV7TfClYaRnn7ORupxy54sdD3mK-eUKb
djNP41qc"
}

(venv) ~/Projects/geistesblitze$
```

Añadir una idea

```
~/Projects/geistesblitze
~/Projects/geistesblitze    ×    ~/Projects/geistesblitze    ×
(venv) ~/Projects/geistesblitze$ http --json --auth eyJhbGciOiJIUzI1NiIsImlhdCI6MTUzNjYyOTIxNywiZXhwIjoxNTM2NjI5ODE3fQ.eyJpZCI6M
X0.5E4CV7TfClYaRnn7ORupxy54sdD3mK-eUKbdjNP41qc:unused :5000/api/ideas/ name=blitzidea description=primera
HTTP/1.0 201 CREATED
Content-Length: 52
Content-Type: application/json
Date: Tue, 11 Sep 2018 01:28:07 GMT
Server: Werkzeug/0.14.1 Python/3.6.5

{
    "description": "primera",
    "id": 1,
    "name": "blitzidea"
}

(venv) ~/Projects/geistesblitze$
```

Actualizar una idea

```
~/Projects/geistesblitze
~/Projects/geistesblitze    ×    ~/Projects/geistesblitze    ×
(venv) ~/Projects/geistesblitze$ http --json --auth eyJhbGciOiJIUzI1NiIsImlhdCI6MTUzNjYyOTIxNywiZXhwIjoxNTM2NjI5ODE3fQ.eyJpZCI6M
X0.5E4CV7TfClYaRnn7ORupxy54sdD3mK-eUKbdjNP41qc:unused PUT :5000/api/ideas/1 name=blitzidea description=segunda
HTTP/1.0 201 CREATED
Content-Length: 52
Content-Type: application/json
Date: Tue, 11 Sep 2018 01:29:52 GMT
Server: Werkzeug/0.14.1 Python/3.6.5

{
    "description": "segunda",
    "id": 1,
    "name": "blitzidea"
}

(venv) ~/Projects/geistesblitze$
```

Listar las ideas

```
~/Projects/geistesblitze
~/Projects/geistesblitze    ×    ~/Projects/geistesblitze    ×
(venv) ~/Projects/geistesblitze$ http --json --auth eyJhbGciOiJIUzI1NiIsImlhdCI6MTUzNjYyOTIxNywiZXhwIjoxNTM2NjI5ODE3fQ.eyJpZCI6M
X0.5E4CV7TfClYaRnn7ORupxy54sdD3mK-eUKbdjNP41qc:unused GET :5000/api/ideas/
HTTP/1.0 200 OK
Content-Length: 110
Content-Type: application/json
Date: Tue, 11 Sep 2018 01:31:56 GMT
Server: Werkzeug/0.14.1 Python/3.6.5

[
    {
        "description": "segunda",
        "id": 1,
        "name": "blitzidea"
    },
    {
        "description": "segunda",
        "id": 2,
        "name": "idearelampago"
    }
]

(venv) ~/Projects/geistesblitze$
```

Obtener los detalles de una idea

```
~/Projects/geistesblitze
~/Projects/geistesblitze        ~/Projects/geistesblitze
(venv) ~/Projects/geistesblitze$ http --json --auth eyJhbGciOiJIUzI1NiIsImlhdCI6MTUzNjYyOTIxNywiZXhwIjoxNTM2NjI5ODE3fQ.eyJpZCI6M
X0.5E4CV7TfClYaRnn7ORupxy54sdD3mK-eUKbdjNP41qc:unused GET :5000/api/ideas/2
HTTP/1.0 200 OK
Content-Length: 56
Content-Type: application/json
Date: Tue, 11 Sep 2018 01:32:45 GMT
Server: Werkzeug/0.14.1 Python/3.6.5

{
    "description": "segunda",
    "id": 2,
    "name": "idearelampago"
}

(venv) ~/Projects/geistesblitze$
```

Eliminar una idea

```
~/Projects/geistesblitze
~/Projects/geistesblitze        ~/Projects/geistesblitze
(venv) ~/Projects/geistesblitze$ http --json --auth eyJhbGciOiJIUzI1NiIsImlhdCI6MTUzNjYyOTIxNywiZXhwIjoxNTM2NjI5ODE3fQ.eyJpZCI6M
X0.5E4CV7TfClYaRnn7ORupxy54sdD3mK-eUKbdjNP41qc:unused DELETE :5000/api/ideas/2
HTTP/1.0 204 NO CONTENT
Content-Length: 0
Content-Type: text/html; charset=utf-8
Date: Tue, 11 Sep 2018 01:33:26 GMT
Server: Werkzeug/0.14.1 Python/3.6.5


(venv) ~/Projects/geistesblitze$
```

Las opciones para la interfaz (en Javascript) que consumirá los servicios REST son numerosas, desde soluciones que "están de moda" como Angular[3], React[4] y Vue.js[5] hasta soluciones más simples como jQuery[6].

Inicialmente pensé en usar Angular o Vue.js, un poco para satisfacer mi propia curiosidad, pero al ver lo necesario para tener un entorno funcionando, antes de escribir una sola línea de Javascript me hizo buscar una solución más simple: jQuery.

Referencias

[1] https://github.com/nnrcschmdt/geistesblitze

[2] https://httpie.org/

[3] https://angular.io/

[4] https://reactjs.org/

[5] https://vuejs.org/

[6] https://jquery.com/



AtixLibre

Hacia un Futuro Innovador

